Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

CRT-2004-06

A SIMPLE COOPERATIVE MULTILEVEL ALGORITHM FOR THE CAPACITATED MULTICOMMODITY NETWORK DESIGN

by

Teodor Gabriel Crainic
Ye Li
Michel Toulouse

March 2004

Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal QC H3C 3J7 Canada
Téléphone : (514) 343-7575 / Télécopieur : (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

HEC · ECOLE POLYTECHNIQUE MONTRÉAL · Université de Montréal

PUBLICATION
CRT-2004-06

A SIMPLE COOPERATIVE MULTILEVEL ALGORITHM FOR THE CAPACITATED MULTICOMMODITY NETWORK DESIGN

by

Teodor Gabriel Crainic [1]
Ye Li [2]
Michel Toulouse [2]

The research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT).

1. Département de management et technologie, Université du Québec à Montréal, and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 (theo@crt.umontreal.ca)
2. Department of Computer Science, University of Manitoba, 545 Machray Hall, Winnipeg, Manitoba, Canada R3T 2N2 (yeli@cs.umanitoba.ca; toulouse@cs.umanitoba.ca)

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2004

Abstract

We describe a first multilevel cooperative cycle-based tabu search for the capacitated multicommodity network design problem. Main design challenges are associated to the specification of each search thread in cooperation, as well as to the definition of the cooperation operators. The paper proposes a first approach to address these challenges and tests it on a set of well-known benchmark problems. The proposed method appears competitive, particularly when difficult problems with high numbers of commodities are considered. Directions and challenges for future research are identified and discussed.

Key words: parallel search, multilevel cooperation, cycle-based tabu search, capacitated multicommodity network design.

Résumé

Cet article décrit un premier algorithme de recherche parallèle coopérative multiniveaux, basé sur des recherches avec tabous par des cycles, pour le problème de synthèse de réseaux multiproduits avec coûts fixes et capacités. La spécification des problèmes traités par chaque programme dans la coopération, ainsi que la définition des opérateurs de coopération comptent parmi les défis les plus sérieux associés à la conception d'un tel algorithme. Nous proposons une première approche que nous testons à l'aide d'un ensemble de problèmes présents dans la littérature. La méthode proposée offre des résultats intéressants, surtout pour les problèmes plus difficiles, avec un nombre élevé de produits. Des directions et défis de recherche sont également identifiés et discutés.

Mots clés : recherche parallèle, coopération multiniveaux, recherche avec tabou par des cycles, synthèse de réseaux multiproduits avec coûts fixes et capacités.

104-04871/

1 Introduction

The capacitated multicommodity network design (CMND) is a well-known NP-hard problem (Magnanti and Wong 1986; Minoux 1986). Exact solution methods are therefore generally excluded when investigating practical approaches to address most CMND problem instances and heuristics are often the methodology of choice (Balakrishnan, Magnanti, and Mirchandani 1997; Crainic 2000; Gendron, Crainic, and Frangioni 1998; etc.).

Parallel computing offers the possibility to design procedures that search more efficiently the solution space of complex problems. This may be achieved through the acceleration of some tedious computational phases of the algorithm, the decomposition of the problem domain or search space, or the design of a multi-thread parallel meta-heuristic that combines several exact or heuristic searches with various degrees of synchronization and cooperation. Multi-thread parallel meta-heuristic search methods should also be more robust than sequential methods with respect to the characteristics of the problem instances they encounter. Parallel implementations are thus expected not only to identify better solutions but also to do it consistently over diverse sets of problem instances without excessive calibration efforts. Cooperative multi-thread strategies appear to offer both the most promising avenues for superior performances and the biggest challenges in terms of algorithm design. Details on parallel meta-heuristic strategies may be found in the reviews of Crainic (2004), Crainic and Toulouse (1998, 2003), Cung *et al.* (2002), Holmqvist, Migdalas, and Pardalos (1997), Pardalos *et al.* (1995), and Verhoeven and Aarts (1995).

Cooperation strategies require careful design, however, in particular with respect to the information exchange mechanism (Toulouse, Crainic, and Gendreau 1996; Toulouse, Crainic, and Sansó 2004; Toulouse *et al.* 1998; Toulouse, Crainic, and Thulasiraman 2000). This is particularly critical when designing algorithms for problems that, like the network design problems we address, combine a combinatorial nature to complex evaluation subproblems (capacitated multicommodity minimum cost flow problems, in the present case). Toulouse, Thulasiraman, and Glover (1999; see also Toulouse, Glover, and Thulasiraman 1998) proposed a new cooperation mechanism that proved very successful for graph partitioning problems (Ouyang *et al.* 2000, 2002). The mechanism is called *multilevel cooperative search* and is based on the principles of local interactions among cooperating searches and controlled diffusion of information.

The goal of this paper is to explore the development of a multilevel cooperative search algorithm for the capacitated multicommodity network design problem, where the individual search threads involved in cooperation make use of the cycle-based

tabu search proposed by Crainic, Gendreau, and Ghamlouche (2003), one of the currently best meta-heuristics for the CMND. We aim to identify the main design challenges and propose a first approach to address them. Experimental results on a set of well-known benchmark problems are presented and analyzed. The proposed method appears competitive, particularly when difficult problems with high numbers of commodities are considered. Directions and challenges for future research are identified.

The paper is structured in the following manner. In Section 2, we recall the mathematical formulation of the CMND problem, briefly survey the literature, and describe the cycle-based tabu search method. Section 3 presents brief descriptions of issues related to the design of parallel meta-heuristics, cooperation mechanisms, and the multilevel method. Section 4 is dedicated to the multilevel method for the CMND problem. We identify and discuss the main the design issues and describe a specific implementation. Section 5 reports computational results, presents a first analysis of the behavior of the multilevel cooperative method applied to the CMND problem, and indicates a number of challenging research directions. We conclude in Section 6.

2 Network Design and Cycle-based Tabu Search

This section is dedicated to a brief overview of the network design formulation and the proposed solution approaches. We also recall the cycle-based tabu search that forms the search method in the cooperative multilevel procedure described later on.

2.1 Fixed Cost, Capacitated, Multicommodity Network Design

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a network with sets of nodes $\mathcal{N}$ and directed arcs $\mathcal{A}$. Without loss of generality, we assume that all $(i, j) \in \mathcal{A}$ are design arcs. Denote f_{ij} and u_{ij}, the fixed cost and the capacity of arc (i, j), respectively. Let $\mathcal{P}$ denote the set of commodities. For each commodity p, one must move w^p units of flow from its (unique) origin $o(p)$ to its (unique) destination $s(p)$. Denote c_{ij}^p the (variable) cost of moving one unit flow of commodity p on arc (i, j).

Two sets of decision variables are defined. *Design* variables y_{ij}, $(i, j) \in \mathcal{A}$, that equal 1 if arc (i, j) is selected in the final design (and 0 otherwise), and *distribution* variables x_{ij}^p indicating the amount of flow of commodity $p \in \mathcal{P}$ on arc (i, j). The

arc-based formulation of the CMND can then be written as

$$\text{Minimize} \quad Z(x,y) = \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{p\in\mathcal{P}}\sum_{(i,j)\in\mathcal{A}} c_{ij}^p x_{ij}^p \tag{1}$$

$$\text{subject to} \sum_{j\in\mathcal{N}^+(i)} x_{ij}^p - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^p = \begin{cases} w^p & \text{if } i = o(p) \\ -w^p & \text{if } i = s(p) \\ 0 & \text{otherwise.} \end{cases} \quad \forall i \in \mathcal{N}, \forall p \in \mathcal{P}, \tag{2}$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p \leq u_{ij}y_{ij} \quad \forall (i,j) \in \mathcal{A}. \tag{3}$$

$$x_{ij}^p \geq 0 \quad \forall (i,j) \in \mathcal{A}, \forall p \in \mathcal{P}. \tag{4}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in \mathcal{A}. \tag{5}$$

where $\mathcal{N}^+(i)$ and $\mathcal{N}^-(i)$ represent the sets of successor and predecessor nodes of node i, respectively. The objective function (1) accounts for the total system cost computed as sum of the fixed costs of the arcs included in a given design plus the cost of routing the product demand on the resulting network. Constraints (2) represent the network flow conservation relations. The linking constraints (3) state that the total flow (all commodities) on an open arc ($y_{ij} = 1$) cannot exceed its capacity, while it must be 0 if the arc is closed ($y_{ij} = 0$). Relations (5) and (4) are the usual non-negativity and integrality constraints for decision variables.

Recall that, for a given design vector $\bar{y}$, the arc-based formulation of the CMND becomes a capacitated multicommodity minimum cost flow problem (CMCF)

$$\text{Minimize} \quad Z(x(\bar{y})) = \sum_{p\in\mathcal{P}}\sum_{(i,j)\in\mathcal{A}(\bar{y})} c_{ij}^p x_{ij}^p \tag{6}$$

subject to the network flow conservation constraints (2) and

$$\sum_{p\in\mathcal{P}} x_{ij}^p \leq u_{ij}\bar{y}_{ij} \quad \forall (i,j) \in \mathcal{A}(\bar{y}),$$

$$x_{ij}^p \geq 0 \quad \forall (i,j) \in \mathcal{A}(\bar{y}), \forall p \in \mathcal{P},$$

where $\mathcal{A}(\bar{y})$ stands for the set of arcs corresponding to the design $\bar{y}$. A solution to the CMND may then be viewed as a design decision $\bar{y}$, assigning a value of 0 or 1 to each design variable, plus the optimal flow $x^*(\bar{y})$ of the corresponding CMCF. The objective function value is then computed accordingly:

$$Z(\bar{y}, x^*(\bar{y})) = \sum_{(i,j)\in\mathcal{A}(\bar{y})} f_{ij}\bar{y}_{ij} + Z(x^*(\bar{y})). \tag{7}$$

The CMND problem is hard and, for now, exact methods cannot solve realistically-dimensioned instances (Crainic, Frangioni, and Gendron 2001; Gendron, Crainic, and

Frangioni 1998; Holmberg and Yuan 2000), even though interesting developments are being proposed (Kim *et al.* 1999; Armacost, Barnhart, and Ware 2002; Chouman, Crainic, and Gendron 2002). For capacitated problems of any interesting size, only specially tailored heuristics have proved of any help. The simplest of these are drop and add procedures based on reduced cost calculations that determine the marginal value of including or excluding an arc from the network (Powell 1986, Koskosidis, Powell, and Solomon 1992), while more sophisticated approaches make use of the marginal values of paths (Crainic and Rousseau 1986, Farvolden and Powell 1994, Jarvis and Mejia de Martinez 1977). These heuristics were applied to particular problem classes, however, and did not attempt to explore past the first local optimum.

Meta-heuristics based on exploring very large neighbourhoods have recently been proposed. Crainic, Gendreau, and Farvolden (2000) proposed a tabu search meta-heuristic for the path-based formulation of the problem that explores the space of the path-flow variables by using pivot-like moves and column generation. Even though the method produced very impressive results, its search efficiency may be limited by the fact that each move considers the impact of changing the flow of one commodity only (a pivot from one path to another). Moreover, one cannot guarantee that all paths will be considered, neither explicitly, nor implicitly. A cooperative search method, based on this procedure and global exchanges through a central memory, has also been proposed and displayed improved performance compared to the sequential search Crainic and Gendreau 2002).

The cycle-based neighbourhood structures for the CMND proposed by Ghamlouche, Crainic, and Gendreau (2003) explore the space of the arc design variables by re-directing flow around cycles and closing and opening design arcs accordingly. These neighbourhoods are part of what appears to be the current best meta-heuristics for the CMND, the path relinking procedure proposed by Ghamlouche, Crainic, and Gendreau (2004). The Lagrangian relaxation-based slope scaling heuristic proposed by Crainic, Gendron, and Hernu (2004) opens interesting perspectives but does not alter the previous statement. Cycle-based neighbourhoods are therefore used in the cooperative method we propose and the basic ideas are recalled in the next subsection.

2.2 Cycle-based Neighbourhoods and Tabu Search

The basic neighbourhood defines moves that may explicitly take into account the impact on the total design cost of potential modifications to the flow distribution of several arcs and commodities simultaneously. The fundamental idea is that one may move from one solution to another by 1) Identifying two points in the network together with two paths connecting these points, thus closing a cycle; 2) Deviating the

total flow from one path to another such that at least one currently open arc becomes empty; 3) Closing all previously open arcs in the cycle that are empty following the flow deviation and, symmetrically, opening all previously closed arcs that now have flow. Such neighbourhoods are huge, however, and their explicit and exhaustive exploration is not practical in most situations. Moreover, the complete evaluation of any design modification involves the resolution of a capacitated multicommodity network flow problem, which rapidly becomes extremely computation intensive. Thus, in order to select the best move for a given solution, the method implements an efficient procedure that avoids the complete evaluation of every examined move and generates a limited number of cycles that include "good" moves.

The approach is based on the observation that not all cycles are of equal interest. One seeks, in particular, moves that modify the status of several arcs and that lead to a significant modification of the flow distribution, such as moves that close at least one arc and open new paths for a group of commodities. To close an arc (i, j), one must be able to deviate all its flow. Let the *residual capacity* of a cycle denote the maximum flow one can deviate around the cycle. The residual capacity of any cycle that includes (i, j) must then be at least equal to $\sum_{p \in \mathcal{P}} x_{ij}^p$, the total flow on arc (i, j). Consequently, the cycles of interest are those that display a residual capacity equal to one of the values in Γ defined as the set of the total (strictly positive) volumes on the open arcs of the corresponding network: $\Gamma(\bar{y}) = \left\{ \sum_{p \in \mathcal{P}} x_{ij}^p > 0 \ : \ (i, j) \in \mathcal{A}(\bar{y}) \right\}$.

Cycles are thus identified on *residual networks* of $\mathcal{A}(\bar{y})$ defined according to $\Gamma(\bar{y})$. The best move in the neighbourhood of $\bar{y}$ is then identified by the cycle leading to the network modification that yields the largest improvement (smallest deterioration, eventually) in the objective function (1). To reduce the computational burden, cycles are identified and evaluated for a set of candidate links. The "lowest" cost cycle for each candidate link is identified by an optimization heuristic based on a modification of the shortest path label-correcting algorithm that avoids getting trapped in negative directed cycles. The method thus progressively builds a set of good candidate neighbours (cycles) among which the best move is selected.

The tabu search-based local search procedure proposed by Ghamlouche, Crainic and Gendreau (2003) integrates two types of cycle-based neighbourhoods. The first type corresponds to the description above and considers the flow of all commodities when determining cycles. Let $\mathcal{C} \subseteq \mathcal{A}$ denote a set of *candidate* links, that is, each arc $(i, j) \in \mathcal{C}$ will be considered as the starting point of cycles. Let $\mathcal{C}(\gamma) \subseteq \mathcal{C}$ include all arcs $(i, j) \in \mathcal{C}$ that can support a movement of γ units of flow. A closed arc may belong to $\mathcal{C}(\gamma)$ only if its capacity is at least γ, while for an open arc, either its flow or its residual capacity must be at least γ units. Such neighbourhoods are identified as *broad*. The second type of neighborhood is identified as *local* and aims to refine the search by implementing moves that result from the deviation of the

flow of one commodity at a time. Local neighbourhoods are used to intensify the search and are implemented by instantiating the set Γ for each commodity, denoted $\Gamma^p = \left\{x^p_{ij} > 0 \ : \ (i,j) \in \mathcal{A}(\bar{y})\right\}$.

Initialization. Generate an initial feasible solution and set *BestSolution* and *CurrentSolution* to its objective value.

Main local search loop. While a stopping criterion is not met

1. Determine sets $\mathcal{C}$, $\Gamma(\bar{y})$, and $\mathcal{C}(\gamma)$ for the current solution $\bar{y}$;
2. Determine the best cycle over all $(i,j) \in \mathcal{C}(\gamma)$, $\gamma \in \Gamma(\bar{y})$

   For each $\gamma \in \Gamma(\bar{y})$

   Build the γ-residual network;

   For each arc $(i,j) \in \mathcal{C}(\gamma)$

   find the lowest cost cycle using the network optimization procedure;

   Select the best move;
3. Move to the new solution by opening and closing the appropriate arcs;
4. Evaluate the new *CurrentSolution* by solving exactly the associated CMCF;
5. Assign a tabu status to each complemented arc;
6. Trim (close arcs with no flow);
7. If the solution in infeasible, perform a *restoration* phase;
8. Perform an *Intensification* phase if current solution is a "good":

$$\frac{CurrentSolution - BestSolution}{BestSolution} \leq IntensGap;$$

9. If $CurrentSolution < BestSolution$ update *BestSolution*.

Figure 1: Tabu Search with Cycle-based Neighbourhoods

The meta-heuristic procedure is synthesized in Figure 1. Following an initialization phase, the procedure explores the search space defined by the design variables using a simple local search framework with tabu tags. At each iteration, the best non-tabu move in a broad neighbourhood is determined and implemented, whether it improves the overall solution or not. A short-term tabu memory is used to record characteristics of visited solutions to avoid cycling. Such broad moves may generate unfeasible solutions detected through the use of artificial arcs. A restoration phase is undertaken in such cases, by restricting the sets $\mathcal{C}$ and Γ to the artificial arcs with positive flow. When a particularly good solution is encountered, the search is inten-

sified using the local neighbourhood and accepting improving moves only. A solution is considered particularly good when it either improves the best overall solution or is close to it by at least a pre-defined percentage. An initial feasible solution is produced by opening all arcs and solving the corresponding flow problem. All arcs with flow are then considered open, while all unused arcs are closed. An intensification phase is then performed until negative cycles are no longer detected for any commodity. The method terminates whenever a predefined stopping criterion (number of iterations, CPU time, etc.) is met. Details and extensive computational results maybe found in Ghamlouche, Crainic and Gendreau (2003).

3 Multilevel Cooperative Search

Cooperative multi-thread search methods are parallel meta-heuristics built upon independent search threads that exchange information. There are essentially two basic approaches to sharing information among threads, through either *global* or *local interactions*.

Global interactions rely on explicit, or approximate, representations of the state of the global search to identify and coordinate the sharing of information. Thus, for example, threads may stop at some global synchronization points where best solutions are exchanged (generally, not a very successful strategy; Crainic and Toulouse 2003). Alternatively, some central depository is used where information, including but not restricted to best solutions, is stored and, possibly, processed, and from where it may be fetched at any time by one of the cooperating search threads. The *adaptive* (Rochat and Taillard 1995) and *central* (Crainic, Toulouse, and Gendreau 1997; Crainic and Gendreau 2002) memory strategies belong to this class of approaches and have been successful addressing a number of difficult problems (e.g., the surveys of Crainic and Toulouse 2003 and Crainic 2004).

Cooperative search procedures based on local interactions are simpler to implement. No global point of reference is used by such methods, neither explicit ones such as synchronization schemes or global control by "higher" algorithms, nor implicit such as a central depositories for exchanged information. In fact, in its most usual form, exchanges of information based on local interactions take place between two search threads: a sender and a receiver. In its simplest form, the exchange consists for the sender to send a solution that is used by the receiver to restart its search from the point in the search space defined by the received solution.

It is important to notice, however, that other than the formal exchange of information taking place between two locally interacting threads, a broader self-organizing

(self-regulating) system of information exchanges is taking place in cooperative multi-thread searches based on local interactions. This system supports the diffusion of information among many threads through long chains of correlated local interactions and feedback loops (Toulouse, Crainic, and Thulasiraman 2000; Toulouse, Crainic, and Sansó 2004). This diffusion process impacts in complex ways the exploration of the solution space performed by cooperating search threads and appropriate mechanisms must be devised to control its behaviour.

The study of Toulouse, Crainic, and Thulasiraman (2000) shows, in particular, that the same values for subsets (blocks) of decision variables are propagated across several search threads through chains of correlated local interactions. (A thread B that has already interacted with another thread A will eventually interact with a third thread C. Assuming threads A and B were senders, then the two local interactions overlap because they share a common thread B. Two local interactions $A - B$ and $B - C$ are *correlated* if interaction $B - C$ occurs because of the occurrence of the local interaction $A - B$.) This study also shows that such duplication of information is a form of spontaneous global control structure that coordinates the exploration of the search space performed by cooperating search threads. Unfortunately, most cooperative search algorithms are not designed to monitor the blocks that get copied. Consequently, when "poor" sets of blocks get widely propagated, this impacts negatively the performance of the overall search because its focus is narrowed to poor regions of the solution space.

The *Multilevel cooperative search* method is a paradigm to design cooperative multi-thread algorithms based on local interactions (Toulouse, Thulasiraman, and Glover 1999; Ouyang *et al.* 2002). It aims to increase the control on the complex information diffusion process generated by chains of correlated local interactions. The multilevel cooperative search mechanism addresses the previously-identified issues by making explicit in the design of cooperative algorithms the process by which blocks of decision variables are formed. Moreover, multilevel cooperative searches also use blocks of decision variables as a mean to define moves, and to decompose the search space to allow parallel explorations. This second characteristic aims to design global control structures that emulate the spontaneous behavior observed in cooperative search procedures (Toulouse, Crainic, and Thulasiraman 2000)

In the next two subsections, we describe the multilevel cooperative search paradigm. We first focus on how block of decision variables are used to define moves and neighborhoods, on the one hand, and to define parallelism, on the other hand. We then describe how blocks of decision variables are selected such that the inherent self-regulation mechanism of cooperative procedures can be exploited for, hopefully, better performances. We conclude with a display of the general structure of a multilevel cooperative search algorithm.

3.1 Moves, Neighborhood Structures, Levels

The multilevel cooperative search concepts have been applied first to the graph and hypergraph partitioning problems (Toulouse, Thulasiraman, and Glover 1999; Ouyang *et al.* 2000, 2002). We use the graph partitioning problem to illustrate the various ideas and definitions.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, $\mathcal{P} = \{1, 2, \ldots, p\}$ the set of p partitions of $\mathcal{G}$. Define x_{ij} the decision variable that takes value 1 if vertex $i \in \mathcal{V}$ belongs to partition $j \in \mathcal{P}$, and 0 otherwise, and let $\mathcal{X}$ be the solution space. Assume we define a procedure that proceeds as follows: "For a feasible solution $x \in \mathcal{X}$, aggregate pairs of consecutive vertices $u, v \in \mathcal{V}$ such that $u < v$ and u, v are in the same partition $j \in \mathcal{P}$". The procedure generates *blocks* of two decision variables with the same value, i.e., the two decision variables x_{uj} and x_{vj} that assign two vertices u and v to the same partition j. We may then define a move that transfers all variables in a block to a different partition and specify a search algorithm that explores the solution space of the graph partitionning problem based on the associated neighborhood.

Many aggregation rules are possible. Toulouse, Thulasiraman, and Glover (1999) and Ouyang *et al.* (2000, 2002) present the complex rules used to for the graph and hypergraph partitioning problems. More simply, one could change how nodes are selected (e.g., $u, v \in \mathcal{V}$ and the same partition $j \in \mathcal{P}$ such that $u < v$ and u, v are even numbers) or the number of nodes that make up a block. Whatever the strategy used to aggregate nodes and thus define blocks, neighborhoods based on transfers or exchanges of blocks between partitions will be different, inducing different searches through the solution space of the graph partitioning problem. These different neighborhood structures may then be viewed as the source for parallelism.

Multilevel cooperative searches are parallel algorithms that use this source of parallelism. The name *multilevel* is derived from the technique used to differentiate neighborhoods, motivated by the goal of controlling the dispersion of information. Thus, the multilevel cooperation mechanism is defined by a number of special information exchange operators, introduced in the next subsection, and a hierarchical neighborhood structure for the global, cooperative search. The hierarchy is obtained by defining blocks of increasing size for different neighborhood structures. No aggregation is used at level 0, the block size thus corresponding to a single decision variable. The number of variables in a block increases with each level, the highest level corresponding to blocks with the largest size.

The *coarsening factor*, that is, the number of decision variables per block, is a parameter. A typical approach would specify the block size at level i to encompass 2^i decision variables. The neighbors of a solution at level i are then all the solutions that

can be reached by changing simultaneously the values of all the decision variables in the blocks involved in a single move. The treatment applied to variables in a block, aggregation, variable fixing, and so on, is application specific. We therefore refer to variables in a block as displaying the same *status*.

3.2 Local Interactions and the Cooperative Search

Local interactions in a multilevel cooperative search procedure take place only between adjacent levels in the hierarchical neighborhood structure. Interactions thus always consist in information being sent by a *sender* search process to a *receiver* search thread that is exactly one level up or down in the hierarchy. This limits the diffusion of information and simplifies to a great extend how information propagates among the cooperating search threads. Two main types of local interactions are defined and implemented through specific operators.

The first type of interaction does not modify the current blocks of the receiver thread, but it modifies (replaces in the extreme case) the solution from which the search will continue. Assuming the receiver thread is at level i, an *interpolation* operator is associated to information that arrives from a sender *above*, at level $i+1$, while a *reverse interpolation* operator is used when information arrives from the level *below* $(i-1)$. The solutions sent from level $i+1$ or $i-1$ are selected among a set of *elite solutions*, defined as the best solutions ever visited by the search thread at the corresponding level. Once a solution is received, the receiver search thread, at level i, modifies its current solution and thus continues its search in a region of the solution space selected strongly based on the solution received from its neighbor.

Notice that, because the neighborhood structures are different at different levels, the search at the receiver level will proceed along a different trajectory from that followed by the sender even when the receiver completely replaces its current solution with the new one and restarts the search. In particular, the neighborhood structure at level i has more degrees of freedom compared to level $i+1$: the blocks of decision variables are smaller and the neighborhood of each solution is larger. Also notice that restarting using an elite solution from the level above is reminiscent of initiating an *intensification* search in the region identified by the elite solution. On the other hand, the neighborhood structure at level i has less degrees of freedom compared to level $i-1$: blocks are coarser and neighborhoods are smaller. A restart using an elite solution from the level below may then be viewed as a somewhat limited diversification of the search from the region identified by the received elite solution.

The second category of local interactions controls the generation of new blocks of

decision variables. These local interactions use information from level $i-1$ to change the blocks at level i. Two local interaction operators were defined for changing blocks in the graph partitioning applications: a *partitioning* operator and a *re-coarsening* operator. The former destroys existing blocks, while the latter creates new ones. By changing blocks at a given level i, some regions of the solution space become unreachable for the search thread at that level, while new ones become available. New blocks yield new neighborhoods and search threads may then explore new regions of the solution space. Such interactions may therefore be considered major *diversification* moves.

A *frequency* vector at level $i-1$ governs the application of the two operators at level i. This frequency vector registers the state of the decision variables among the best solutions visited (the set of elite solutions) by the search thread at level $i-1$. Decision variables that appear with the same values in several elite solutions are considered "settled". In terms of the search method at level i, these variables can be handle as if they were a single decision variable. Therefore, decision variables at level $i-1$ that have similar frequency values form new blocks at level i. Similarly, blocks at level i are dismantled if they are composed of two or more blocks from level $i-1$ with frequency values that are substantially different from one another.

Graph and hypergraph partitioning problems have been the object of intensive research for quite some time now (e.g., Kernighan and Lin 1970), and several multilevel algorithms have been proposed (e.g., Hendrickson and Leland 1993; Cong and Shinnert 2003). Yet, the application of the principles summarized above, combined to a good local search heuristic (Fiduccia and Matteyses 1982) applied at each level, has produced superior results (Ouyang *et al.* 2002). We hope to reproduce this success for the CMND problem.

Chains of correlated local interactions and the propagation of blocks of decision variables constitutes inherent mechanisms of self-regulation in cooperative procedures based on local interactions. The multilevel cooperative search approach proposes a way to influence this spontaneous behavior by controlling which blocks are formed and by having elite solutions drive the constitution of these blocks. The macroscopic dynamics of the parallel search is thus more likely to proceed under the control of an optimization logic. This is an important achievement for the multilevel algorithm design compared with other cooperative algorithms. Furthermore, the use of blocks of decision variables to define moves, neighborhood structures, and to obtain parallelism is a nice elucidation of the spontaneous structuring process taking place in cooperative procedures as blocks of decision variables get duplicated across several search threads.

4 Multilevel Algorithm for the CMND Problem

This section is dedicated to the presentation of the multilevel cooperative algorithm proposed for the CMND problem. We start with the definition of levels and the specification of the initial values. The local interaction operators are defined next. The algorithm is summed up in the last subsection.

4.1 Level Definition

The first issue one must address when contemplating the development of a multilevel cooperative search algorithm for the CMND problem is how to define levels. Recall that the main idea is to define levels by aggregating decision variables. On the other hand, one has to preserve the characteristics of the initial problem to ensure the correctness of the diffused information.

It thus become clear quite rapidly that a straightforward aggregation of the nodes (and arcs) of the network, similar to that practiced for the graph partitioning problem, is not appropriate for the CMND. On the one hand, node aggregation makes the definition of the resulting link and, especially, node fixed costs and capacities is awkward at best. On the other hand, while responding to the goal of the graph partitioning problem, graph aggregation does not relate to the objectives of a design process: deciding which arcs to select and which ones to discard.

We therefore define levels by the number of arcs that are *fixed*, each fixed arc being either *fixed-open* or *fixed-closed*. An arc that is fixed cannot have its status changed by the tabu search process that proceeds at the corresponding level. In fact, a fixed-closed arc "disappears" from the graph, while a fixed-open arc "looses" its fixed cost (appropriate accounting has to be enforced, of course) and cannot be closed. Only interactions with neighboring levels may yield a modification in the fixed status of arcs. Thus, the fixed status of an arc inhibits the implementation of all cycles involving a change in its status and reduces the number of neighbors in the corresponding search space. The specification for each level of the fixed-open and fixed-closed arcs thus directly impacts the exploration of the solution space performed by the tabu search program.

To create the initial set of levels, one needs to fix the size of this set, the number of levels, as well as the coarsening factors for each level specifying the number of arcs that are fixed-open and fixed-closed at each level. One also needs a procedure to select which arcs become fixed-open and fixed-closed at each level. The number of levels and the coarsening factors are parameters with values obtained through calibration

(Section 5).

The strategy to select the initial arcs in fixed-open and fixed-closed blocks should aim to select arcs that are likely to have the the status "open" or "closed" in good network designs. We considered two methods to estimate this likelihood. The first makes use of the FC ratio defined as $\frac{fixed\ cost}{capacity}$. A small FC ratio may be interpreted as a signal that the corresponding arc might be open in good designs since it brings capacity to the network with a relatively limited increase in fixed cost. Symmetrically, a large FC ratio is taken to indicate that the corresponding arc may not appear in good designs.

The second heuristic consists in selecting blocks of decision variables based on an initial exploration of the solution space. In this case, the multilevel cooperative search starts as a parallel independent search made up of l independent sequential cycle-based tabu search procedures, where l is the number of levels in the multilevel procedure. Each independent search uses the same initial solution (all arcs are open). Because the cycle-based tabu search selects neighbors randomly, the independent searches eventually diverge in their exploration of the solution space.

To each level i is associated a set of *elite solutions* as well as a *frequency memory*. The elite set contains the best solutions found at level i. The size of this set is a pre-defined parameter. The frequency memory is an integer vector V of size equal to the number of decision variables. Each entry $V[a]$ records the number of times the corresponding decision variable y_a, $a = (i, j)$, $i, j \in \mathcal{A}$, had the status open in solutions with a cost no greater than some percentage of the worst solution in the elite set.

The initial blocks of fixed-open and fixed-closed arcs for each level i are created once the cycle-based tabu search has failed to improve the best known solution for a pre-defined number of iterations. At that point, the coarsening factors pre-selected for level i are used to determine the size of each block. The decision variables with the highest frequency values in the frequency memory become fixed-open while those with the lowest frequency values become fixed-closed.

Once the blocks are generated at all levels, the search proceeds according to the multilevel cooperative idea: at each level the cycle-based tabu search attempts to improve the current solution given the blocks of fixed variables, while local interactions are executed by each level with its neighbor levels above and below in the multilevel structure. Search threads proceed asynchronously. At each level, the information required for cooperating with neighbor levels, the elite set and the frequency memory, is available at all time, therefore a level does not have to wait for the neighbor levels to initiate cooperation.

4.2 The Local Interaction Operators

Several local interaction operators have been considered to support cooperation among levels. Three have been retained: *interpolation*, *re-coarsening*, and *reverse interpolation*. We felt that a specific partitioning operator is not needed for the CMND problem since the two interpolation operators, the reverse interpolation one in particular, indirectly destroy blocks.

The *interpolation operator* is used to replace the current solution of the sequential cycle-based tabu search at level i based on an elite solution from level $i+1$ and start the search in the new region identified by the imported solution. The interpolation operation is initiated from level i by requesting one of the solutions in the elite set of level $i+1$ that has not already been sent to level i. The open (closed) status of some of the decision variables in the imported solutions from level $i+1$ may conflict with the status of some of the fixed decision variables at level i, however. Thus, for example, some arcs may be part of the network design in the solution from level $i+1$, while these same arcs are fixed-closed at level i. In order to restart the search at level i using the imported elite solution from level $i+1$, the fixed-closed status is canceled for all the arcs that conflict with the imported solution. This makes sure that the search in level i will proceed freely in the region identified by the imported elite solution. Removing the fixed status of arcs at a given level has the effect of a partitioning operator in this sense that it increases the number of available moves (cycles) in the neighborhood of solutions at level i. Denote Y_i the vector $Y_i[a]$ that gives the status of arc a at level i. If $Y_i[a] = 1$, then arc a is part of the fixed-open variables. If $Y_i[a] = 0$, a is part of the fixed-closed variables. If $Y_i[a] = 2$, arc a is a design variable. The pseudo-code of the interpolation operator is given in Figure 2.

```
Interpolation(Y_i);
  Get an elite solution ȳ_{i+1} from level i + 1
  for (a = 1; a ≤ n; a++)
    if arc ȳ_{i+1}[a] is opened then Y_i[a] = 2;
  Return Y_i;
  Restart search at level i with ȳ_{i+1} as initial solution;
```

Figure 2: Interpolation Operator

The *reverse interpolation* operator is similar to the interpolation operator, except that it propagates information **up**, from lower to higher levels. It uses the same elite set as the interpolation operator. At regular intervals, level i requests from level $i-1$ the best elite solution that has not yet been sent to level i. Upon receiving this elite

solution, the fixed-closed status is removed at level i for all arcs that conflict with their status in the imported elite solution. Similar to the interpolation operator, reverse interpolation potentially reduces the size of the fixed-closed block, which increases the density of the neighborhood structure at level i. Figure 3 describes this operator using pseudo-code.

```
Reverse_Interpolation(Y_i);
  Get an elite solution ȳ_{i-1} from level i − 1
  for (a = 1; a ≤ n; a++)
    if arc ȳ_{i-1}[a] is opened then Y_i[a] = 2
  Return Y_i;
  Restart search at level i with ȳ_{i-1} as initial solution;
```

Figure 3: Reverse Interpolation Operator

The *re-coarsening operator* proceeds as follows. At regular intervals, i.e., after a predetermined number of iterations of the cycle-based tabu search method, level $i = 1, \ldots, l-1$ makes a request to obtain a copy of the frequency memory V_{i-1} from level $i-1$. The blocks of fixed-open and fixed-closed decision variables at level i are then modified based on the frequency memory V_{i-1}. Decision variables with a high frequency in V_{i-1} have their status changed to fixed-open while those with low frequency have their status changed to fixed-closed. The re-coarsening operator is also used to bring the number of arcs with fixed status in line with the figures associated to the coarsening factors for level i.

The re-coarsening operator is described in Figure 4. The input variables cfo_i and cfc_i represent the block sizes of fixed-open and fixed-closed variables, respectively, according to the coarsening factors for level i. The frequency memory V_{i-1} is scanned to sort decision variables in decreasing order of their frequency values (the **for** loop of line 1). $B[k] = p$ indicates that decision variable p has the k^{th} highest frequency. The vector Y_i is then updated according to the frequency memory V_{i-1} (**for** loops of lines 2 and 3).

4.3 The Multilevel Algorithm for the CMND Problem

The three interaction operators, interpolation, reverse interpolation, and re-coarsening, support cooperation among the cycle-based tabu search programs. Each level of the multilevel algorithm executes the sequence of steps describes in Figure 5. Each level explores the solution space according to the neighborhood structure imposed by the

```
Re-coarsening(Y_i, cfo_i, cfc_i):
  int A, C, B[1..n];
  get the frequency vector V_{i-1} from level i − 1
1. for (k = 1; k ≤ n; k++)
     A = 0;
     for (p = 1; p ≤ n; p++)
       if ((V_{i-1}[p] ≥ A)&(V_{i-1}[p] ≠ ∞)) then
         A = V_{i-1}[p];
         C = p;
     B[k] = C; V_{i-1}[C] = ∞;
2. for (k = 1 to cfc_i)
     Y_{i+1}[B[k]] = 1;
3. for (k = n − 1 downto cfo_i)
     Y_i[B[k]] = 0;
  return Y_i;
```

Figure 4: The Re-coarsening Operator

two blocks of fixed decision variables. The iterations of the cycle-based tabu search procedures are interrupted to communicate with adjacent levels using one the three interaction operators. Interactions are local and take place asynchronously. All the tabu procedures are identical for all the levels except for level 0 and 1. For these two levels, tabu search performs intensification phases that help to explore more thoroughly the regions of the solution space identified as "promising" by higher levels. Our stopping criterion is a pre-defined number of cycle-based tabu search iterations, including the iterations executed to obtain the initial blocks of fixed-open and fixed-close arcs. The number of tabu iterations is the same for all levels.

Steps 2, 4, and 6 state the sequence of calls to specific interaction operators. This particular sequence has been set using the following logic: The first interaction operator is a restart of the cycle-based tabu search program at level i, $i = 0, \ldots, l-2$ using an elite solution from level $i+1$. This arrives, in principle, after the cycle-based tabu has exhausted its search at level i. With reference to level $i+1$, the search that follows an interpolation is a form of intensification performed with the help of the less coarsened neighborhood structure of level i. Once the tabu search has explored the region of the solution space identified by the interpolation operator, step 4 calls the re-coarsening operator. The re-coarsening operator offers a second chance to explore the same region identified by interpolation with a new neighborhood structure provided by the frequency memory of level $i-1$. Step 6 is the last step in the sequence of calls to interaction operators. This is also a restart, but using an

Generate the initial sets of fixed-open and fixed-closed blocks;

while (stopping criterion not satisfied) **do**

1 Perform cycle-based tabu search iterations;

2 If $i \neq l-1$, perform an *interpolation*;

3 Perform cycle-based tabu search iterations;

4 If $i \neq 0$, perform *re-coarsening*;

5 Perform cycle-based tabu search iterations;

6 If $i \neq 0$, perform a *reverse interpolation*;

Figure 5: Core of the Multilevel Cooperative Algorithm for the CMND Problem

elite solution from the lower level. Chances are that the imported elite solution from level $i-1$ is a local optimum or that it comes from the basin of attraction from which the search program at level $i-1$ failed to escape. The restart of this exploration in the more coarsened neighborhood of level i should help to cross barriers toward promising neighbor regions in the solution space. With reference to level $i-1$, the search that follows a reverse interpolation is a form of diversification performed with the help of a more coarsened neighborhood structure.

5 Computational Results

This section is dedicated to the presentation and analysis of the results of the experiments performed with the simple multilevel cooperative design presented in the previous section. This is the first study of a multilevel strategy applied to the CMND problem. Moreover, it represents one of the very few studies so far of the multilevel parallel strategy for meta-heuristics. Hence, in analyzing the results, we focus on the capability of the multilevel design to yield high quality solutions for the CMND problem and on the understanding of the multilevel cooperation mechanism. This explains the somewhat lengthier than usual discussion on the calibration results as well as the subsection on the search behavior of the multilevel algorithm.

To ensure meaningful comparisons, we experiment on the same problem instances used by Ghamlouche, Crainic, and Gendreau (2003, 2004). Problem instances are general transshipment networks with no parallel arcs. Each commodity corresponds to a single origin-destination pair. On each arc, routing costs are the same for all commodities. Problem instances have been generated to offer for each network size a

variety of fixed-to-routing-cost and capacity-to-demand ratios. A detailed description of problem instances is given in Crainic, Frangioni, and Gendron (2001; see also Gendron and Crainic 1994, 1996). The problem generators as well as the problem instances can be obtained from the authors.

The multilevel program has been coded in C++. The exact evaluation of the capacitated multicommodity network flow problems is performed using the linear programming solver of CPLEX version 7.5. The parameters used for the cycle-based tabu search are those indicated in Ghamlouche, Crainic, and Gendreau (2003). Tests were conducted on a Sun Enterprise 10000 with 64 400 MHz clock processors and 64 Gb of RAM. The parallel programming model for the multilevel procedure is shared memory using Pthreads under Solaris 2.7.

5.1 Parameter Calibration

Initial experiments have been performed to determine values for a number of key parameters and design features. Calibration tests have been conducted using eight problem instances with 20 nodes and 300 design arcs with different number of commodities (40 and 200) and various combinations of fixed-versus-variable-cost ratio and capacity tightness measures. Our previous work on developing meta-heuristics for design problems indicates that these problems offer a reasonable sample of characteristics and difficulty. The following parameters have been studied using these problem instances: 1) Number of levels; 2) Coarsening factors, that is, the size of the fixed-open and fixed-closed blocks at each level; 3) Number of tabu iterations between activations of the interaction operators; 4) Dimension of the elite solutions set; 5) Updating of the frequency memory; 6) Selection of initial arcs to be fixed.

The number of levels and the coarsening factors are closely related. A large number of levels increases the level of parallelism and, potentially, the performance of the global search. When coarsening factors increase substantially from one level to the next, the corresponding dimension of the solution space tends to decrease rapidly, however. Consequently, a large number of levels comes at the cost of low coarsening factor values. On the other hand, low coarsening factor values imply small blocks of fixed decision variables which conducts to weak interactions among search threads and weak impact of parallelism on the way searches are performed at the various levels of the structure. This is a behavior pattern that one aims to avoid. Thus a trade-off between large coarsening factor values and a high number of levels is usually sought.

This trade-off is particularly difficult to find for CMND problems. Thus, increas-

ing the number of fixed variables rapidly yields problems that are infeasible due, in particular, to the interactions between demand and capacity and the associated commodity competition for limited transportation capacity. Moreover, the issue is still relevant when the problem dimensions are increased. Indeed, the number of decision variables (design variables in the present case) is a determining factor for the number of levels and the values of the coarsening factors and, in general, higher values may be used for these two parameters when the number of decision variables is increased. Unfortunately, most CMND problems of interest display the same behavior relative to the demand versus capacity factor independently of their dimension in number of design arcs (problem instances with almost no slack capacity or, at the other extreme, with almost unlimited capacity are significantly easier to solve).

We have decided therefore that, for this first study, we will fix the number of levels for all problem instances based on the number of design decision variables in the smallest of the problem instances tested (100 arcs). We aimed to select a number of levels that minimizes overlapping between parallel search threads in the solution space. Experimentally, eight levels appeared as the maximum level of parallelism we could achieve for those instances and were used for all calibration and experimentation activities.

To calibrate the coarsening factors, one must consider that fixed-open and fixed-closed arcs do not have the same impact on the exploration of the solution space. During the neighborhood exploration performed by the cycle-based tabu search, cycles that contain fixed-closed arcs are automatically eliminated. On the other hand, a cycle with fixed-open arcs may generate a move if this move does not involve changes to the status of the fixed-open arcs in the cycle. In other words, a fixed-closed status impacts in a more stringent manner the exploration of the solution space than the fixed-open status. Consequently, we considered smaller block sizes for fixed-closed arcs. We tested increasing the total number of arcs per level by 3%, 6%, and 9%, respectively, and coarsening factors with a ratio 1/3 for the fixed-closed and 2/3 for the fixed-open blocks, respectively. A coarsening factor of 9% per level, with 3% of fixed-closed arcs and 6% of fixed-open arcs, offered the best results for the calibration tests.

For this first multilevel algorithm for the CMND problem, we adopted a simple strategy of activating the local interaction operators after a pre-defined number of cycle-based tabu iterations. We tested the values 5, 10, 20, and 50. A value of 20 offered the best performance on the restricted calibration problem set and has been used throughout the experimentation phase.

The calibration tests indicated that, in general, differences in the cardinality of the elite solution sets did not impact significantly the solution performance. It also

appeared, however, that the cardinality should not be "too high" and that using elite sets that include several solutions dominates elite sets of size one. The rational seems to be the following. When single-elite-solution sets are used, the elite set would be updated significantly less frequently than when larger sets are used. Consequently, there is a high occurrence of the interpolation and reverse interpolation operators sending the same solution to neighbor levels. On the other hand, large elite sets might contain solutions of lower quality that, when transmitted to neighboring levels will send the exploration at those level astray. A cardinality of 4 for the elite sets appeared a reasonable compromise in our calibration results and has been used in the experimentation phase.

The goal of the frequency vectors is to accumulate information about the status of decision variables in good solutions. The same compromise encountered for the elite set cardinality calibration is also present in this case. Restricting "good" solutions to those that are very close (less than 5%, say) to the overall best solution enforces the elite characteristic of the frequency vector, but yields procedures where applications of the re-coarsening operator have little impact on coarsening. On the other hand, the update of the frequency vectors based on "poor"-quality solutions implies driving the search (through the re-coarsening operator) into possibly uninteresting regions of the solution space. A maximum value of 5% less than the cost of the worst solution in the elite set at the corresponding level gave the best performance.

We have also tested the two approaches for determining the initial variables to be fixed described in Section 4.1. The approach based on the frequency memory appeared superior to that based on the *FC* ratio. In the frequency-memory approach, the initial coarsening takes place once the cycle-based tabu search has failed to improve the best known solution for a number of iterations. We tested values of 20, 50, and 100 for this parameter and a value of 100 performed best. We impose, however, a limit (fixed at 200) for the maximum number of iterations the search at each level may run in independent mode. Consequently, a level switches to cooperation mode either when it has failed to improve its best solution during the last 100 iterations or when it has reached the 200-iteration limit.

5.2 Search Patterns on CMND Problem Instances

We first present a first analysis and interpretation attempt of the search behavior of the multilevel cooperative mechanism. We illustrate this analysis by charting the evolution of the parallel search for 3 representative problem instances. For each problem instance, Figures 7, 8, and 9 display the solution value at each iteration for each of the 8 levels, for 600 iterations of the tabu search method. Function values reaching the

upper limit of the graphs indicate infeasible solutions. For comparison purpose, Figure 6 display the same evolution results for a parallel independent algorithm running 8 cycle-based tabu search procedures for 600 iterations.

All methods, sequential, parallel independent, and multilevel, behave in essentially the same way during the early stages of the search. This is because all methods use the cycle-based tabu search and the multilevel search proceeds in an independent way during initialization. It is noteworthy that levels 0 and 1 of the multilevel algorithm display steeper descents due to the intensification phases that help find better solutions.

The multilevel method exhibits a marked drop in solution value between iterations 150 to 250, that is, when the initial blocks of fixed decision variables are generated and inter-level cooperations starts to kick in. This drop may be noticed at all levels, but it is more accentuated at levels 0, 1, and 2 where the search has more degrees of freedom (smaller blocks). The marked improvement brought by the initiation of cooperation is followed by a more gradual improvement in the best-solution value. This behavior may be observed for all problem instances and is a clear indication of the impact of the cooperation mechanism. To illustrate this behavior, we follow the evolution of the search in Figures 8 and 9.

For problem instance C56 (Figure 8), level 0 finds its best initial solution at iteration 51, while running in independent mode. Recall that at level 0 one does not generate blocks of fixed decision variables, nor perform reverse interpolation or re-coarsening local interactions. Consequently, the drop that occurs at iteration 152 of level 0 is due solely to the interpolation operator that imports the best elite solution from level 1. The best initial solution for level 1 is found at iteration 62 and the interactions start at iteration 162: fixed-open and fixed-close blocks are defined, followed by an execution of the interpolation operator with level 2. The drop is not as sharp as for level 0, level 1 imports an elite solution from level 2, which, by observing the chart, is not very good. The other levels ralley later on, mostly between iterations 180 to 240. For problem instance C61 (Figure 9), the best initial solutions for levels 0 and 1 are found respectively at iterations 106 and 119. Cooperation mode for these two levels (as well as for most of the other levels) kick in at iteration 200. Knowing that interactions between levels occur at every 20 iterations, from the chart of problem instance C61, we can easily observe the impact of the cooperation operators at each level. For example, level 1 has sharp drops at iteration 220 and 280 caused both by re-coarsening interactions.

A significant difference may be observed between the independent and the multilevel parallel searches, as illustrated in Figures 6 to 9. Any thread participating in a parallel independent search may identify good solutions, based on its degree of success

in the random selection of neighbors by the cycle-based tabu search method. This is expected as the search threads are independent from one another. The situation is different for the multilevel method, where the lowest levels are almost exclusively the ones where good solutions are found. Thus, in our examples, most improving solutions are found at levels 0 to 2, while levels 3 and 4 sometimes obtain solutions close to the best ones, and levels 6 and 7 contributing very little due to the large number of fixed arcs. This characteristic of the multilevel search follows from the combined effect of local interactions, the hierarchical structure of neighborhoods, and the tabu search intensification phases used at levels 0 and 1. This observation does not mean that higher levels are not useful. On the contrary, they contribute to diversify the search.

A second difference between independent and multilevel searches is apparent in the figures: Functions cross less often each other in multilevel search than for independent search. One may infer than that the multilevel neighborhood structure based on increasing sizes of blocks is relatively successful at preventing overlaps among parallel explorations of the solution space. Moreover, the gradual improvement observed once the cooperation is active supports the claim that cooperation by local interactions among levels is successful in driving levels 0 and 1 towards good regions of the solution space.

5.3 Result Analysis

Table 1 displays the results of the computational experiments. Problem instances are identified in the first column by a number, together with their characterization in terms of the number of nodes, arcs, and commodities, as well as the fixed cost and capacity type: a relatively high or low fixed cost relative to the routing cost is signaled by the letter **F** or **V**, respectively, while letters **T** and **L** indicate respectively if the problem is tightly or somewhat loosely capacitated compared to the total demand. Columns TC and AV.TC hold respectively the best and the average solution out of three repetitions of the cycle-based tabu search executed for 400 iterations (Ghamlouche, Crainic, and Gendreau 2003). Columns PR and AV.PR display the same information for the path relinking approach (Ghamlouche, Crainic, and Gendreau 2004). Column MLEVEL reports the results obtained for one run of the multilevel algorithm for 600 iterations.

The results are very encouraging. Compared to the path relinking, the best current meta-heuristic for the CMND problem, the multilevel cooperative method displays very competitive results. The objective function values of the solutions identified by the multilevel method are often equal or better than those obtained by the path

PROB	TC	PR	AV. TC	AV. PR	MLEVEL
C1(25,100,10,V,L)	14712	14712	14769.33	14712	14712
C2(25,100,10,F,L)	14941	14941	14941	15081.33	14941
C3(25,100,10,F,T)	50529	49899	50619.67	50154	49937
C4(25,100,30,V,T)	365385	365385	365385	365385	365385
C5(25,100,30,F,L)	37515	37654	38672.33	37704	37607
C6(25,100,30,F,T)	87325	86428	88095.33	86492.33	86461.3
C7(100,400,10,V,L)	28786	28485	28836.67	28529	28553
C8(100,400,10,F,L)	24022	24022	24022	24022	24022
C9(100,400,10,F,T)	67184	65278	67376.33	65539.33	66284
C10(100,400,30,V,T)	385508	384926	385512.67	385181.67	385282
C11(100,400,30,F,L)	51831	51325	52176.33	51875.67	50456
C12(100,400,30,F,T)	147193	141359	147478	143403.67	145721
C33(20,230,40,V,L)	430628	424385	430768.33	424789.67	426702
C35(20,230,40,F,T)	652775	645548	657803.67	646378.67	652894
C36(20,230,40,V,T)	372522	371811	372888	372162.33	371475
C37(20,230,200,V,L)	100001	100404	101481.33	101469.33	98582
C38(20,230,200,F,L)	148066	147988	148975	151352	143150
C39(20,230,200,V,T)	106868	104689	107589.33	105598.67	102030
C40(20,230,200,F,T)	147212	147554	147868	148044.33	141188
C41(20,300,40,V,L)	432007	429398	432168.33	429523.67	429837
C42(20,300,40,F,L)	602180	590427	604072.67	594590.67	593544
C4320,300,40,V,T)	466115	464509	466546.33	464822.67	466004
C44(20,300,40,F,T)	615426	609990	616859.67	609995.33	619203
C45(20,300,200,V,L)	81367	78184	82186.67	79095.33	78209.5
C46(20,300,200,F,L)	122262	123484	123247.67	124924.67	121951
C47(20,300,200,V,T)	80344	78866.80	80724	79212.27	77251
C48(20,300,200,F,T)	113947	113584	115342	114632.33	111173
C49(30,520,100,V,L)	56603	54904	56722	55043.67	55754
C50(30,520,100,F,L)	103657	102054	105276.33	104222.33	99817
C51(30,520,100,V,T)	54454	53017	54537	53212	53512
C52(30,520,100,F,T)	105130	106130	107885.33	107575	102477
C53(30,520,400,V,L)	122673	119416	123277.33	119624	115671
C54(30,520,400,F,L)	164140	163112	165458	163377	156601
C55(30,520,400,V,T)	122655	120170	123210.33	120764.33	120980
C56(30,520,400,F,T)	169508	163675	170301.33	164921.33	160217
C57(30,700,100,V,L)	50041	48723	50098.33	48894.33	48869
C58(30,700,100,F,L)	64581	63091	64802.67	63436.33	63756
C59(30,700,100,V,T)	48176	47209	48249.67	47288.33	47457
C60(30,700,100,F,T)	57628	56575.50	58111.33	56808.83	56910
C61(30,700,400,V,L)	107727	105116	108459.33	105403.33	102631
C62(30,700,400,F,L)	150256	145026	150909	147887.33	143988
C63(30,700,400,V,T)	101749	101212	103112.33	102119.33	99194.9
C64(30,700,400,F,T)	144852	141013	146705	141446.67	138266

Table 1: Computational Results

relinking. Moreover, all other problem characteristics being kept constant, the multilevel appears to perform better when the number of commodities is increased (which, normally, increases the difficulty of the problem).

Much work is still needed, however, in order to achieve the best multilevel design for the CMND problem. A number of issues have been highlighted by the experiments. One observes, for example, a relatively high number of infeasible solutions being generated. One also notices that the searches proceeding at high levels (e.g., levels, 5, 6, and 7 in Figures 7 to 9) are not very productive in terms of generating good solutions. Moreover, one may also notice that in many cases the searches at the various levels do not proceed at the same pace, nor do they produce networks with similar designs. We believe that one of the main reasons behind these behaviors is the very (probably, too) simple and stringent definition of the cooperation mechanism, particularly the coarsening operation and factors.

Indeed, the cycle-based neighborhood definition is prone to yield moves that bring the search to infeasible solutions (see the example in Ghamlouche, Crainic, and Gendreau 2003), as illustrated for the independent search of Figure 6. Fixing arcs, particularly fixing them closed, reduces the search space and increases the likelihood of such moves, especially for problem instances with tight total capacity compared to total demand. Then, when high coarsening factors are used, e.g., for higher levels, the search space becomes very constrained and many moves conduct to infeasible solutions. On the other hand, fixing arcs modifies the problem instance and thus the behavior of the tabu search. Thus, when computing cycles, the cost of moving flow on a fixed-open arc is equal to the variable cost, while this cost is weighted by a linearization of the fixed cost when the arc is undecided. This may bias the search towards using the fixed arcs and not allow a thorough exploration of the neighborhood at the corresponding level.

One must also recall that, for CMND problems, the search proceeds in the space of the design variables only. The feasibility of the design resulting from a move as well as the associated value of the design and flow distribution must therefore be computed by solving a rather complex capacitated, multicommodity minimum-cost network flow problem. The impact of aggregating variables in blocks to define levels is therefore much more difficult to determine than, for example, in the case of the graph partitioning problem. One may thus set the search in a not-so-interesting region of the solution space of the original problem and it might take many interactions before moving towards a more interesting region.

A prioritary research direction corresponds therefore to the study of coarsening operators. The ones studied in this paper are quite simple, as they are based on characteristics of individual arcs. More elaborated ones should exploit more the

characteristics of the multicommodity network design problem. Thus, for example, we plan to explore coarsening operators based on the structure of the networks, e.g., paths or trees, and the associated commodity definition. The relations between the problem characteristics, in terms of dimensions and capacity, on the one hand, and the number of levels and coarsening operators and factors, on the other hand, will also be part of this study.

A second, and complementary, research direction concerns the flexibility of the cooperation structure. Currently, the mechanism is quite strict. The operators are invoked at regular intervals, in strict order. The literature on parallel meta-heuristics indicates, however, that introducing flexibility in the algorithms is beneficial. Thus, while the strict mechanism used in the present paper yields very good results, we plan to study more advanced mechanisms that would allow the exploration to dynamically adapt to the problem instance and the state of the search.

Exploring in this direction brings up issues related to the utilization of memories in cooperation. Simple memories are already included in the algorithm and used by the interaction operators. More advanced memories may be built. Indeed, similar to solution values, other solution and exploration attributes may be put into memories and associated to interaction operators. Such data may even be part of the information diffusion process. The utilization of this diffusion to create or approximate at each level an image of the status of the entire search is a fascinating challenge.

6 Conclusions

We presented the first multilevel cooperative parallel tabu search algorithm for the capacitated multicommodity network design problem. Computational results on a set of benchmark problem instances show that this approach yields good quality solutions comparable to those obtained by the current best meta-heuristics for the problem.

These results are very encouraging but they also emphasize the need for more research to understand better the interplay between the CMND problem, the cycle-based tabu search, and the multilevel strategy for cooperative parallel search, and to develop the next generation of algorithms. We are currently undertaking this research and hope to report in the near future.

Acknowledgments

Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada, through its Discovery Grant programs, and by the Fonds F.C.A.R. of the Province of Québec.

While working on this project, Dr. T.G. Crainic was Adjunct Professor at the Département d'informatique et de recherche opérationnelle of the Université de Montréal (Canada), the department of Computer Science of the University of Manitoba at Winnipeg (Canada), and at Molde University College (Norway).

References

Armacost, A.P., Barnhart, C., and Ware, K.A. (2002). Composite Variable Formulations for Express Shipment Service Network Design. *Transportation Science*, 36(1):1–20.

Balakrishnan, A., Magnanti, T.L., and Mirchandani, P. (1997). Network Design. In Dell'Amico, M., Maffioli, F., and Martello, S., editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 311–334. John Wiley & Sons, New York, NY.

Chouman, M., Crainic, T.G., and Gendron, B. (2002). Valid Inequalities for Multicommodity Capacitated Fixed Charge Network Design. Technical report, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.

Cong, J. and Shinnert, J.R., editors (2003). *Multilevel Optimization in VLSICAD*, volume 14 of *Combinatorial Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Crainic, T.G. (2000). Network Design in Freight Transportation. *European Journal of Operational Research*, 122(2):272–288.

Crainic, T.G. (2004). Parallel Computation, Co-operation, Tabu Search. In Rego, C. and Alidaee, B., editors, *Adaptive Memory and Evolution: Tabu Search and Scatter Search*. Kluwer Academic Publishers, Norwell, MA.

Crainic, T.G., Frangioni, A., and Gendron, B. (2001). Bundle-Based Relaxation Methods for Multicommodity Capacitated Network Design. *Discrete Applied Mathematics*, 112:73–99.

Crainic, T.G. and Gendreau, M. (2002). Cooperative Parallel Tabu Search for Capacitated Network Design. *Journal of Heuristics*, 8(6):601–627.

Crainic, T.G., Gendreau, M., and Farvolden, J.M. (2000). A Simplex-Based Tabu Search Method for Capacitated Network Design. *INFORMS Journal on Computing*, 12(3):223–236.

Crainic, T.G., Gendron, B., and Hernu, G. (2004). A Slope Scaling/Lagrangean Perturbation Heuristic with Long-Term Memory for Multicommodity Capacitated Fixed-Charge Network Design. *Journal of Heuristics*.

Crainic, T.G. and Rousseau, J.-M. (1986). Multicommodity, Multimode Freight Transportation: A General Modeling and Algorithmic Framework for the Service Network Design Problem. *Transportation Research B: Methodology*, 20B:225–242.

Crainic, T.G. and Toulouse, M. (1998). Parallel Metaheuristics. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 205–251. Kluwer Academic Publishers, Norwell, MA.

Crainic, T.G. and Toulouse, M. (2003). Parallel Strategies for Meta-heuristics. In F. Glover and G. Kochenberger, editors, *State-of-the-Art Handbook in Metaheuristics*, pages 475–513. Kluwer Academic Publishers, Norwell, MA.

Crainic, T.G., Toulouse, M., and Gendreau, M. (1997). Towards a Taxonomy of Parallel Tabu Search Algorithms. *INFORMS Journal on Computing*, 9(1):61–72.

Cung, V.-D., Martins, S.L., Ribeiro, C.C., and Roucairol, C. (2002). Strategies for the Parallel Implementations of Metaheuristics. In Ribeiro, C. and Hansen, P., editors, *Essays and Surveys in Metaheuristics*, pages 263–308. Kluwer Academic Publishers, Norwell, MA.

Farvolden, J.M. and Powell, W.B. (1994). Subgradient Methods for the Service Network Design Problem. *Transportation Science*, 28(3):256–272.

Fiduccia, C.M. and Matteyses, R.M. (1982). A Linear Time Heuristics for Improving Network Partitions. In *Procedings 19th ACM/IEEE Design Automation Conference*, pages 175–181. IEEE Computer Society Press.

Gendron, B. and Crainic, T.G. (1994). Relaxations for Multicommodity Network Design Problems. Publication CRT-965, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.

Gendron, B. and Crainic, T.G. (1996). Bounding Procedures for Multicommodity Capacitated Network Design Problems. Publication CRT-96-06, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.

Gendron, B., Crainic, T.G., and Frangioni, A. (1998). Multicommodity Capacitated Network Design. In Sansó, B. and Soriano, P., editors, *Telecommunications Network Planning*, pages 1–19. Kluwer Academic Publishers, Norwell, MA.

Ghamlouche, I., Crainic, T.G., and Gendreau, M. (2003). Cycle-based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design. *Operations Research*, 51(4):655–667.

Ghamlouche, I., Crainic, T.G., and Gendreau, M. (2004). Path Relinking, Cycle-based Neighbourhoods and Capacitated Multicommodity Network Design. *Annals of Operations Research*.

Hendrickson, B. and Leland, R. (1993). A Multilevel Algorithm for Partitioning Graphs. Report SAND93-1301, Sandia National Laboratories.

Holmberg, K. and Yuan, D. (2000). A Lagrangean Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem. *Operations Research*, 48(3):461–481.

Holmqvist, K., Migdalas, A., and Pardalos, P.M. (1997). Parallelized Heuristics for Combinatorial Search. In Migdalas, A., Pardalos, P., and Storoy, S., editors, *Parallel Computing in Optimization*, pages 269–294. Kluwer Academic Publishers, Norwell, MA.

Jarvis, J.J. and Mejia de Martinez, O. (1977). A Sensitivity Analysis of Multicommodity Network Flows. *Transportation Science*, 11(4):299–306.

Kernighan, B. and Lin, S. (1970). An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 49:291–307.

Kim, D., Barnhart, C., Ware, K., and Reinhardt, G. (1999). Multimodal Express Package Delivery: A Service Network Design Application. *Transportation Science*, 33(4):391–407.

Koskosidis, Y.A., Powell, W.B., and Solomon, M.M. (1992). An Optimization–based Heuristic for Vehicle Routing and Scheduling with Soft Time Windows Constraints. *Transportation Science*, 26:69–85.

Magnanti, T.L. and Wong, R.T. (1986). Network Design and Transportation Planning: Models and Algorithms. *Transportation Science*, 18(1):1–55.

Minoux, M. (1986). Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications. *Networks*, 19:313–360.

Ouyang, M., Toulouse, M., Thulasiraman, K., Glover, F., and Deogun, J.S. (2000). Multi-Level Cooperative Search: Application to the Netlist/Hypergraph Partitioning Problem. In *Proceedings of International Symposium on Physical Design*, pages 192–198. ACM Press.

Ouyang, M., Toulouse, M., Thulasiraman, K., Glover, F., and Deogun, J.S. (2002). Multilevel Cooperative Search for the Circuit/Hypergraph Partitioning Problem. *IEEE Transactions on Computer-Aided Design*, 21(6):685–693.

Pardalos, P.M., Pitsoulis, L., Mavridou, T., and Resende, M.G.C. (1995). Parallel Search for Combinatorial Optimization: Genetic Algorithms, Simulated Annealing, Tabu Search and GRASP. In Ferreira, A. and Rolim, J., editors, *Proceedings of Workshop on Parallel Algorithms for Irregularly Structured Problems, Lecture Notes in Computer Science 980*, pages 317–331. Springer-Verlag, Berlin.

Powell, W.B. (1986). A Local Improvement Heuristic for the Design of Less-than-Truckload Motor Carrier Networks. *Transportation Science*, 20(4):246–357.

Rochat, Y. and Taillard, É.D. (1995). Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1(1):147–167.

Toulouse, M., Crainic, T.G., and Gendreau, M. (1996). Communication Issues in Designing Cooperative Multi Thread Parallel Searches. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 501–522. Kluwer Academic Publishers, Norwell, MA.

Toulouse, M., Crainic, T.G., and Sansó, B. (2004). Systemic Behavior of Cooperative Search Algorithms. *Parallel Computing*, 21(1):57–79.

Toulouse, M., Crainic, T.G., Sansó, B., and Thulasiraman, K. (1998a). Self-Organization in Cooperative Search Algorithms. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2379–2385. Omnipress, Madisson, Wisconsin.

Toulouse, M., Crainic, T.G., and Thulasiraman, K. (2000). Global Optimization Properties of Parallel Cooperative Search Algorithms: A Simulation Study. *Parallel Computing*, 26(1):91–112.

Toulouse, M., Glover, F., and Thulasiraman, K. (1998b). A Multi-Scale Cooperative Search with an Application to Graph Partitioning. Report, School of Computer Science, University of Oklahoma, Norman, OK.

Toulouse, M., Thulasiraman, K., and Glover, F. (1999). Multi-Level Cooperative Search. In Amestoy, P., Berger, P., Daydé, M., Duff, I., Frayssé, V., Giraud, L., and Ruiz, D., editors, *5th International Euro-Par Parallel Processing Conference*,

volume 1685 of *Lecture Notes in Computer Science*, pages 533–542. Springer-Verlag, Berlin.

Verhoeven, M.G.A. and Aarts, E.H.L (1995). Parallel Local Search. *Journal of Heuristics*, 1(1):43–65.

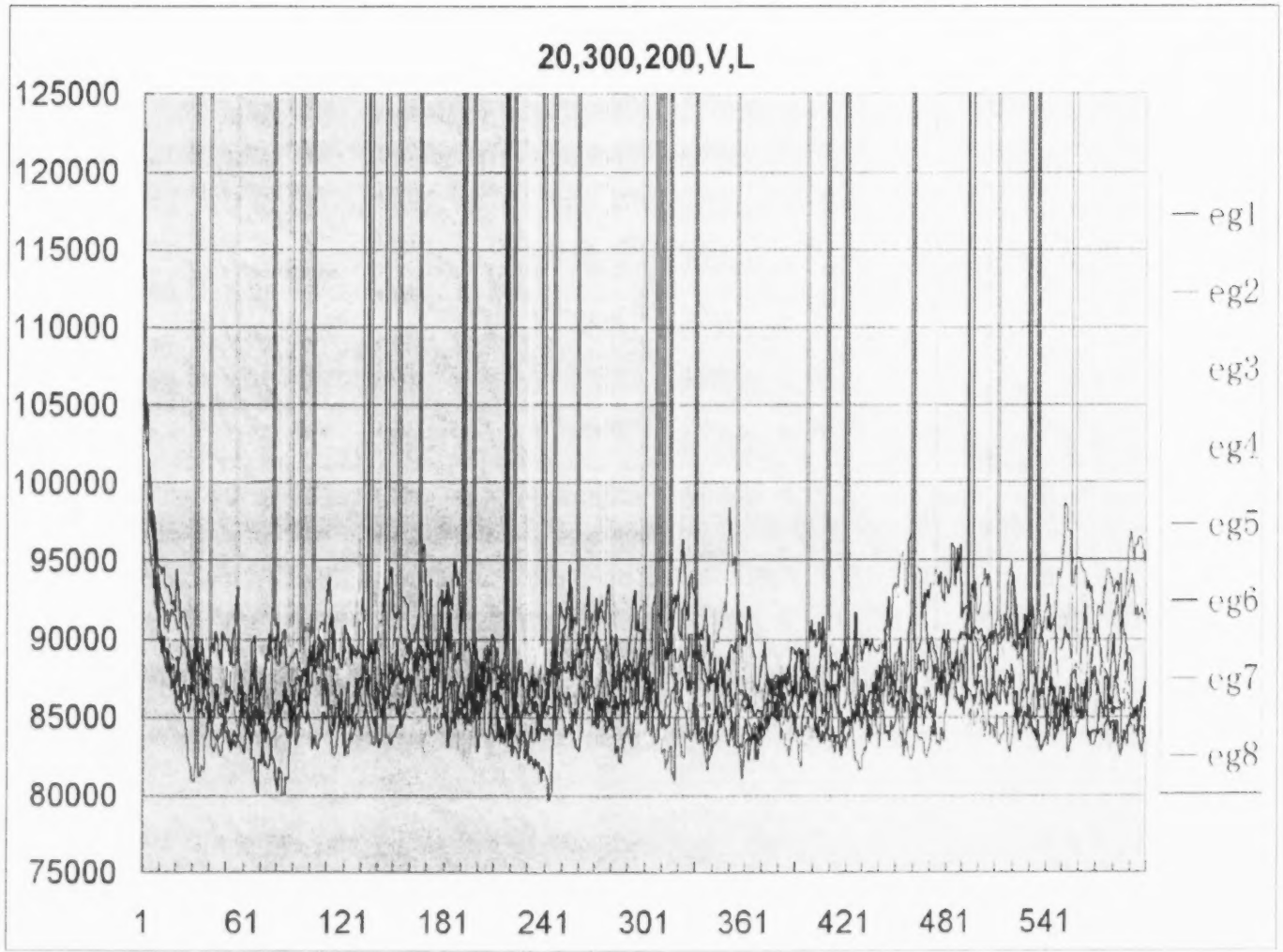

Figure 6: Parallel Independent Search for Problem C45

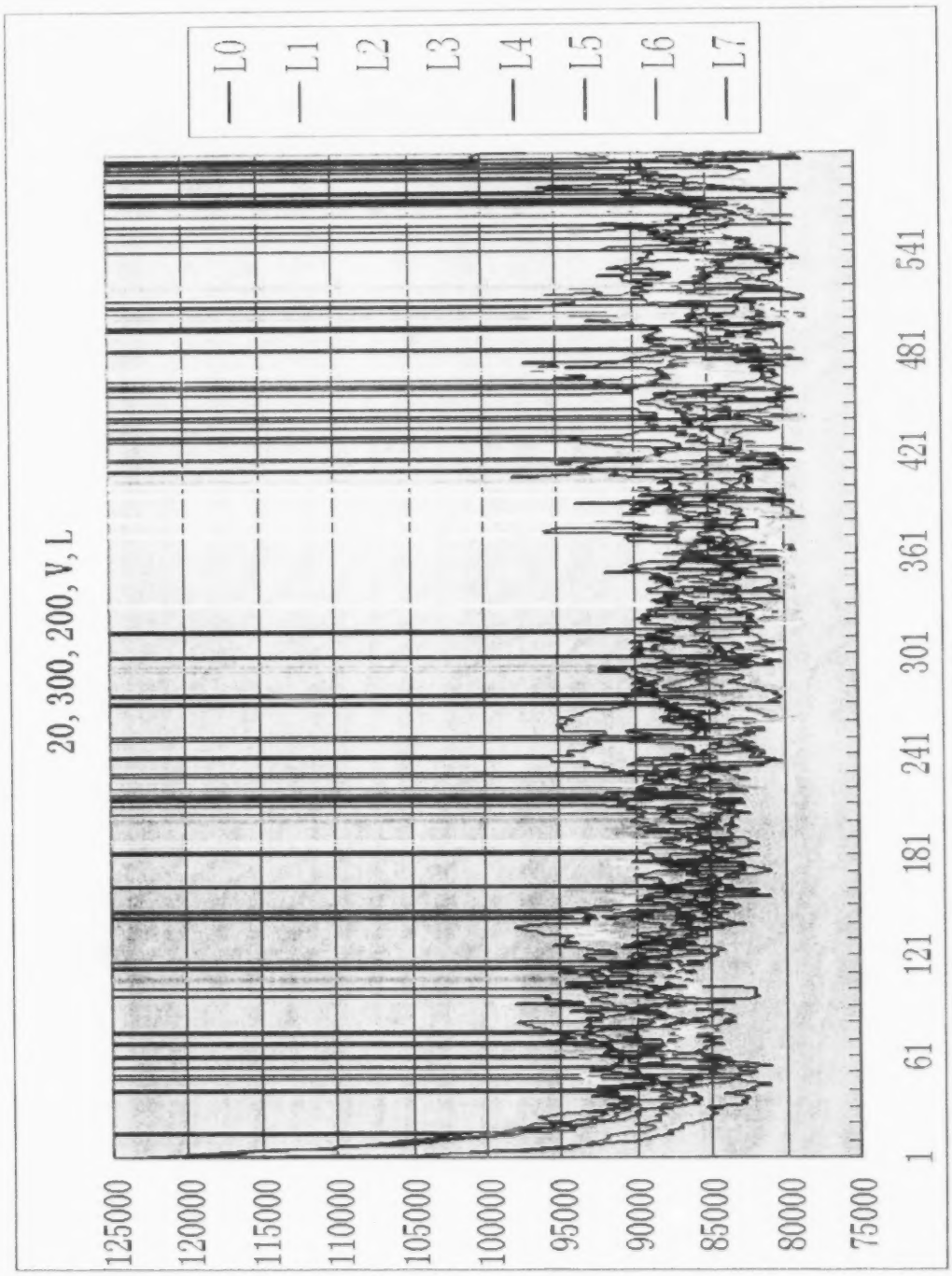

Figure 7: Multilevel Cooperative Search for Problem C45

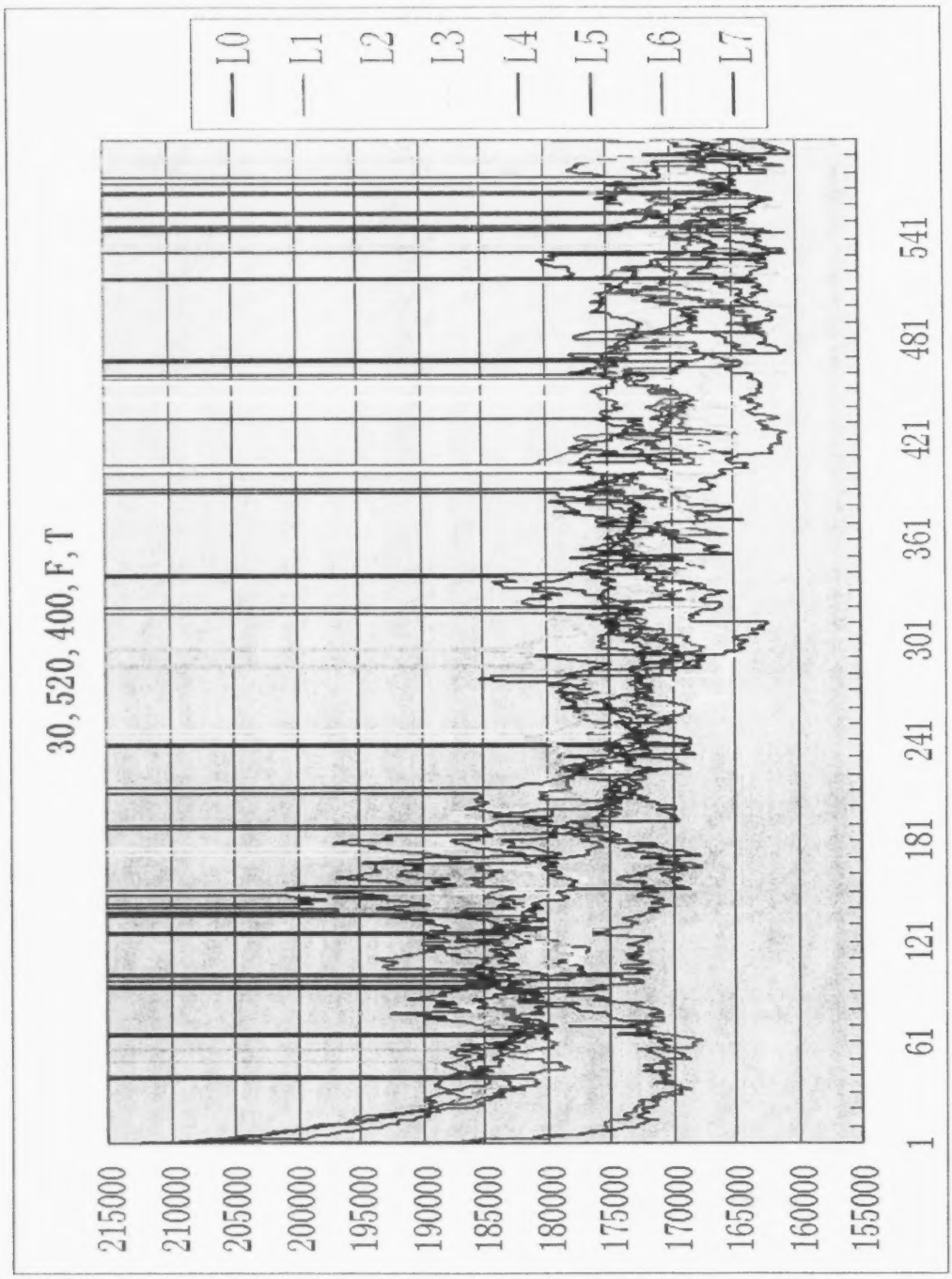

Figure 8: Multilevel Cooperative Search for Problem C56

Figure 9: Multilevel Cooperative Search for Problem CC61

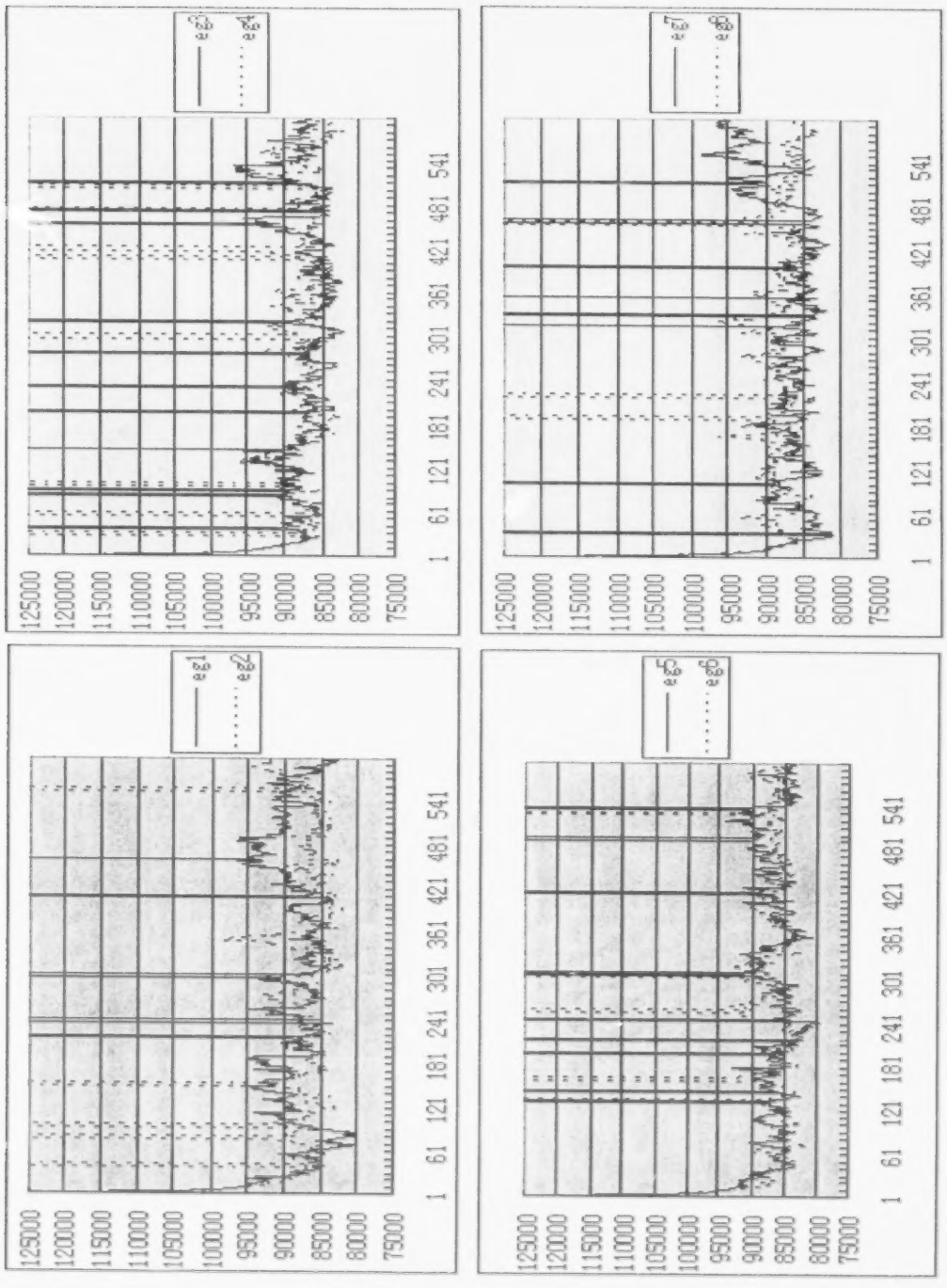

Figure 10: Parallel Independent Search for Problem C45

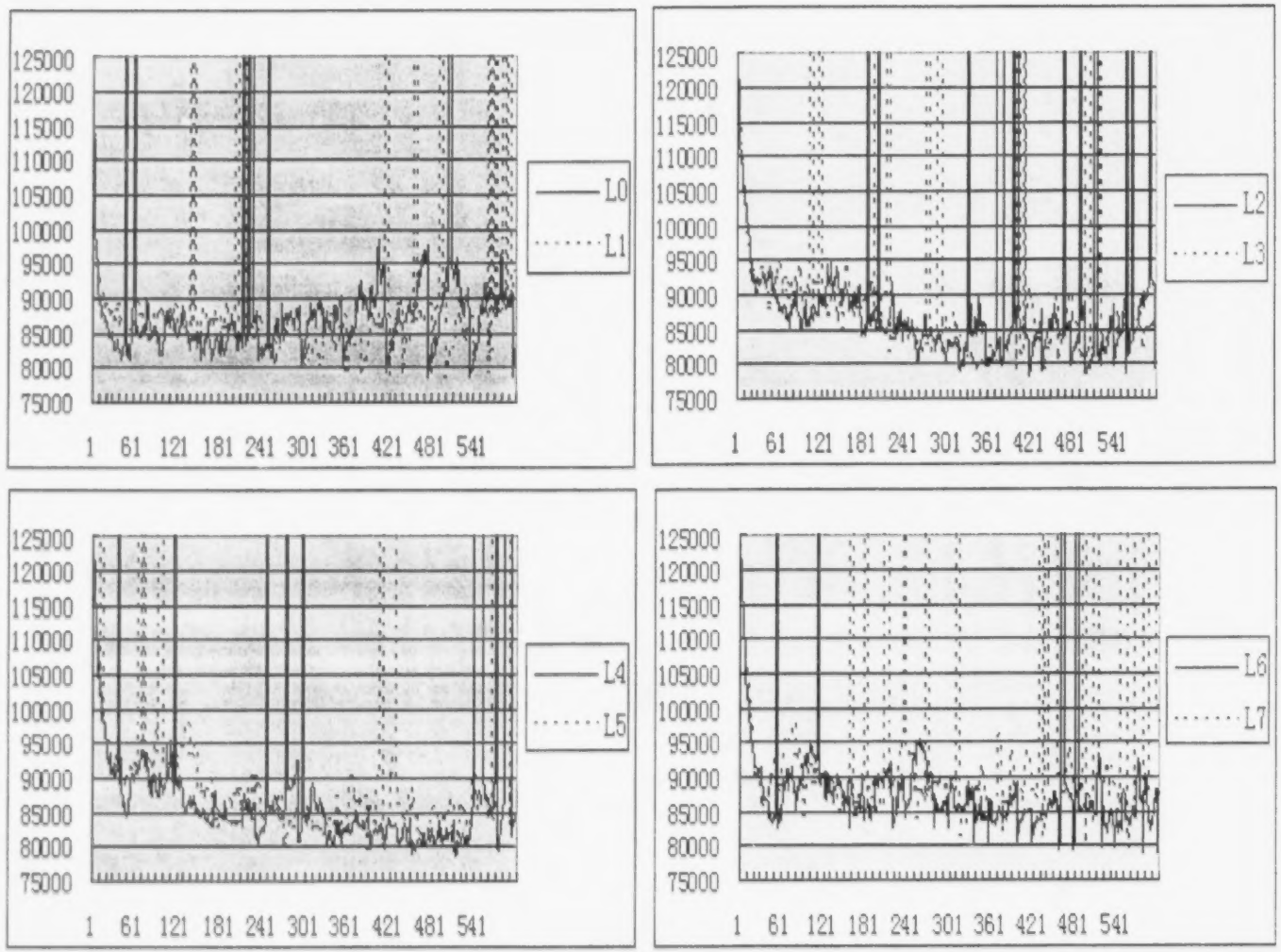

Figure 11: Multilevel Cooperative Search for Problem C45

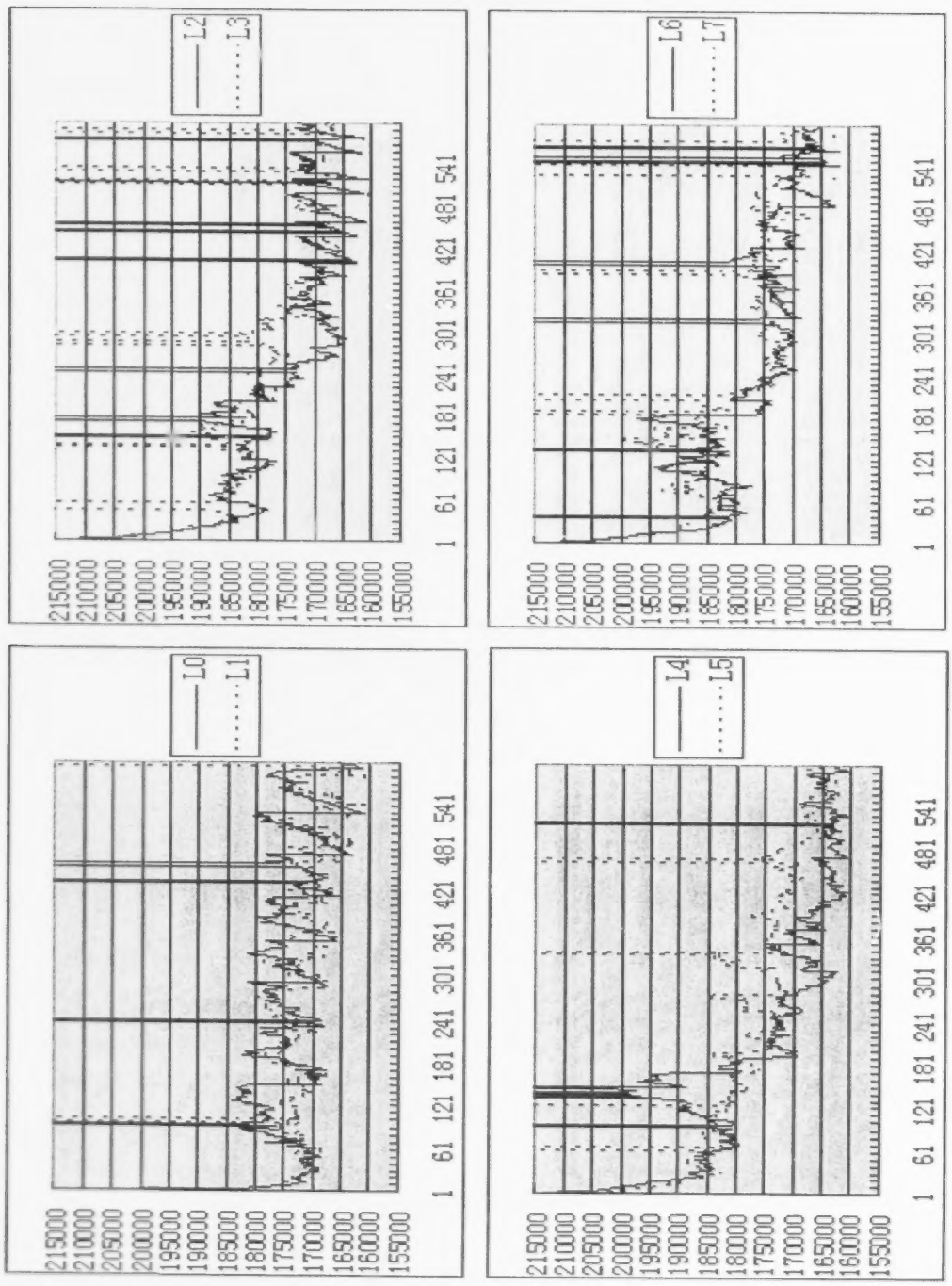

Figure 12: Multilevel Cooperative Search for Problem C56

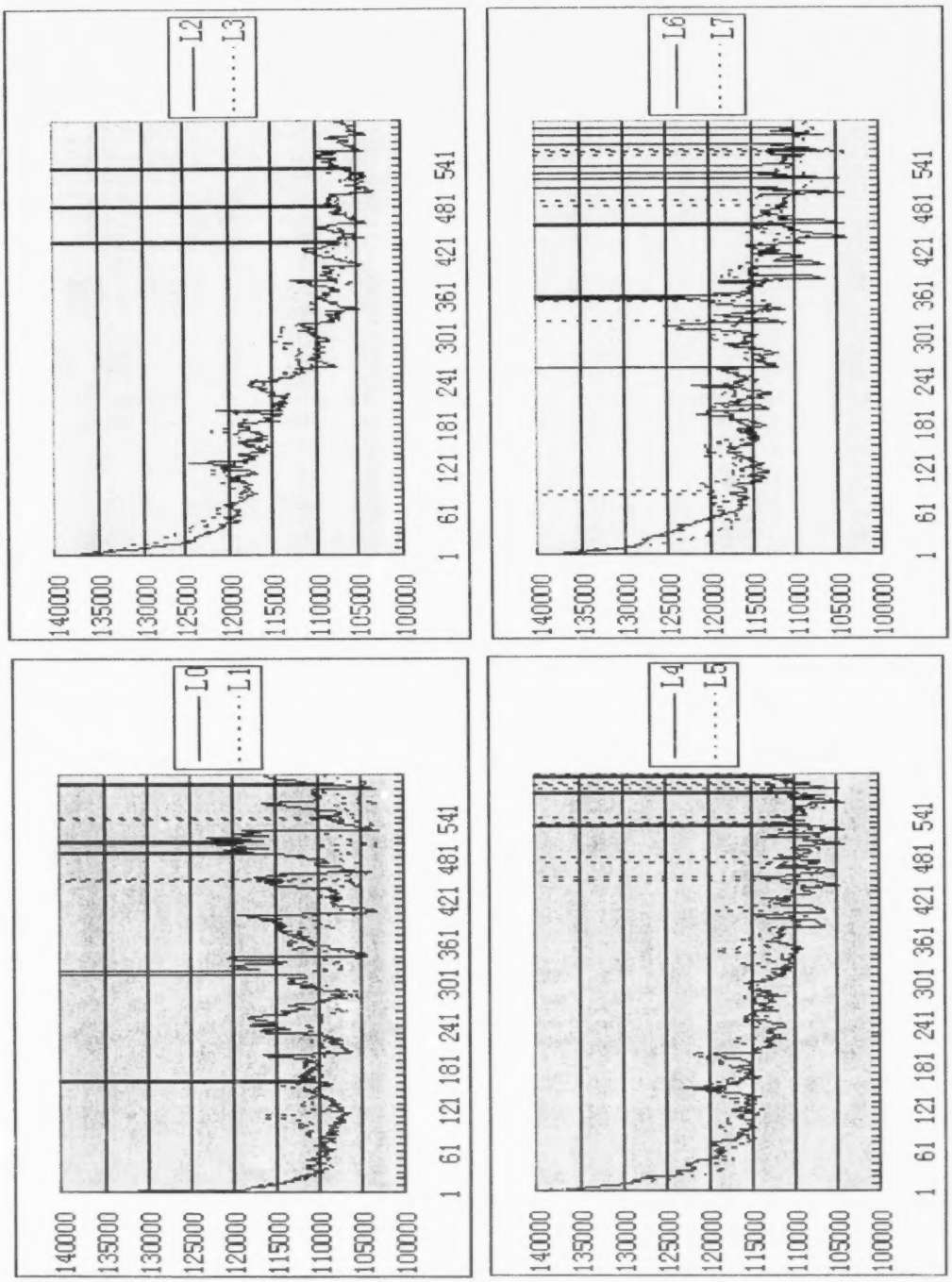

Figure 13: Multilevel Cooperative Search for Problem C61